Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2009, Article ID 323095, 9 pages
doi:10.1155/2009/323095

Research Article

Player Profile Management on NFC Smart Card for Multiplayer Ubiquitous Games

Romain Pellerin,[1,2] Chen Yan,[1] Julien Cordry,[1] and Eric Gressier-Soudan[1]

[1] *CNAM-CEDRIC, 292 rue St Martin, 75141 Paris Cedex 03, France*
[2] *GET-INT, 9 rue Charles Fourier, 91011 Evry Cedex, France*

Correspondence should be addressed to Julien Cordry, julien.cordry@cnam.fr

Received 30 January 2009; Accepted 14 July 2009

Recommended by Zhongke Wu

One of the goals of mixed reality and ubiquitous computing technologies is to provide an adaptable and personal content at any moment, anywhere, and in any context. In Multiplayer Ubiquitous Games (MUGs), players have to interact in the real world at both physical and virtual levels. Player profiles in MUGs offer an opportunity to provide personalized services to gamers. This paper presents a way to manage MUG player profiles on an NFC Smart Card, and proposes a Java API to integrate Smart Cards in the development of MUGs. This user centric approach brings new forms of gameplay, allowing the player to interact with the game or with other players any time and anywhere. Smart Cards should also help improve the security, ubiquity, and the user mobility in traditional MUGs.

Copyright © 2009 Romain Pellerin et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. Introduction

We deeply believe that the next step for the gaming industry will be Multiplayer Ubiquitous Games (MUGs). In this type of game, users play simultaneously in the real world and in the virtual world [1]. To manage an MUG system which supports social interactions among interconnected users in both worlds, the system has to manage the equipments that are deployed in the real world, and to compute the state of the virtual world.

Our purpose here is to enhance the mobility and the ubiquity in MUGs by using a user-centric approach. This might give rise to new kinds of user interactions.

Various technologies, such as RFID tags, networked objects or environmental sensors, can be used to help the user interact with his/her physical environment. Moreover, the players can have access to hand-held device, biomedical sensors, interaction devices, virtual reality glasses, and so forth. Then, various network connectivities are used to link all these devices: Wi-Fi, Bluetooth, ZigBee, or cellular phone networks. Finally, an MUG server could run the global game logic, centralize the game data, and bring the players together. A proper way to support this technological heterogeneity is to use a middleware, like uGASP [2, 3], which is an OSGi-based [4] open-source middleware dedicated to MUGs.

On the gameplay level, MUG systems introduce the concept of Real world Gaming system Interaction (RGI). It is based on the following properties. Firstly, the gameplay relies on the player's physical mobility and often requires a context and a user adaptation. Secondly, the game interacts with the player in an ubiquitous way (at nondedicated locations through nondedicated objects), and proactively (at uncontrolled times, e.g., through email or phone). Finally, the game leads to social interactions which can be effective in the real world or in the virtual world.

So, MUG systems have to be flexible and adaptable enough to be able to respond to these complex and uncertain relations between the real world and the game world, and between the player and the real world. Furthermore, the player should be able to interact with the game despite a network disconnection, for example, to interact with a smart toy in a nonnetworked area. On the design level, like all games, and, more generally, like all entertainment applications, an MUG system should include a user model. An MUG system can be seen as an information system

requiring some user personal data in order to integrate the user's real life into the game, for example, his/her phone number or his/her real life social relations. Natkin and Yan [5] propose a player profile model to provide a personalized gaming experience to the player.

One of the ways to store a player profile in an MUG system is to let the player carry the profile along with him/her on an embedded computing device, such as NFC Smart Card. The Near Field Communication (NFC, [6]) Smart Cards are a fast growing member of the large Smart Card family. Today, Smart Cards are widespread devices with cryptographic and storage capabilities, and tamper-resistant properties. This makes those devices ideal for many application contexts like in identification, transport, telecommunication, or banking domains. Their non-self-powered essence implies the use of a reader/Card Acceptance Device (CAD) that can power up the card and interact with it. The NFC technology enables them to interact with their environment in a contactless manner, most primarily with mobile phones.

No public attempt to manage an MUG player profile on a Smart Card has been provided so far. Besides, it appears that many game systems tend to understate the security and confidentiality issues that should be addressed in any networking environment while some personal data are involved. The work undertaken here is part of the PLUG research project [7]. PLUG is led by the CNAM-CEDRIC computer science research laboratory in collaboration with Musée des Arts et Métiers, Orange Labs, Institut Sud Telecom, L3i lab from University La Rochelle, and a game studio: TetraEdge. It aims at creating an MUG inside the CNAM museum that takes into account the player characteristics. This MUG is built on top of the uGASP middleware.

This paper introduces a user centric approach dedicated to MUG systems. Our approach consists in using an NFC Smart Card to store the MUG player profile, providing mobility, and guaranteeing user privacy and confidentiality. The player holds some game information in order to interact with the surrounding NFC devices. In addition, the Smart Card provides the player with a secure way to store confidential data. In this work, we present an open-source service to manage MUG player profile (MUGPP) for Java-based devices (card, reader, and server levels): the MUGPPM API (the API for our MUG Player Profile Management). Section 2 describes the MUGPP. Section 3 presents the technologies used for user profile management on Smart Cards. Section 4 discusses the benefits of handling the player profile on an NFC Smart Card for MUGs and the kinds of new interactions it could bring to the user and the MUG system. In Section 5, the general architecture of the system is presented and the security issues related to the protection of the data in the card are discussed. Section 6 describes the new kinds of interactions using our API. The last section concludes and gives our perspectives for future work.

2. Player Profile Definition for MUG (MUGPP)

The essence of gameplay is designing a game with regards to the user point of view. This point of view is implicitly or explicitly coded in the game system: all games and all entertainment applications include a user model. In single player games, it starts from a rough classification of the target players and a limited memory of player actions in the game, but it can also be a complex cognitive model. In multiplayer games, the model contains social attributes and behaviors. In multiplayer ubiquitous games, the model has to be cognitive, social, and related to the history and to the current situation of the player in both the virtual and the real world.

Considering that the user's space of activity embeds computing devices and that information systems become more and more ubiquitous and pervasive, there is a need to consider the interaction between the real and the virtual world in a mixed reality mode, and the possible actions of the user in both universes. So the user model will not only take into account the state and behavior of the user as in classical online gaming situations but also in augmented outdoor or mobile gaming environments.

Our method is to use an explicit user model, the MUG Player Profile (MUGPP), to gather and classify distinctive information about the player. This information will be the deductive basis for the game decision mechanism.

The MUGPP guides the game decision engine to offer diverse game experiences to players. The game quests adapt the game scenario to the personal context of the player, which leads to an action that is executed both in the game and in the real world. The main goal in the use of the MUGPP along with the automatic generation of the narration is to decide which type of quest can best relate to the player profile and to the global narration needs, so as to promote social relations between players. In this way, the playability of the game is augmented: the game is persistent and adaptable. Each player can have a unique experience.

The MUGPP depends on a set of parameters that can be either statically defined by the game designer or dynamically adjusted according to the real time changes in the user's physical states or even in the user's social features. It implies a personalized level of parameters in the user model [5]. Since the player is represented in both the real world and the virtual system, we have to consider his/her knowledge of the gameplay from several different points of view. It is very useful to distinguish the user's general information from his/her in-game data, as his/her general profile could be re-exploited by different game mechanisms. The following three groups describe the kinds of user information that are collected and identified.

The first group includes some data about the user "by himself, " that is, unrelated to his/her game practice: civil status, preferences, and so forth. Most of this data can only be provided directly by the player during the creation of the game account. Since this data changes infrequently, it has to be accessible by any MUG on the game platform, so the player does not need to register his/her civil status every time he/she plans to play a new game.

The second group collects the knowledge about the user defined "as a player." It includes some exact information corresponding to the basic choices of the player: the type of account, distribution of the duration of play in each location, and so forth. It includes also statistical data or some real-time

data gathered during the play: his/her physical location, his/her interaction with the various interactive devices in the real environment, and so forth.

The third group defines the status of the player's avatar in the game from both a statistical and a real-time point of view, such as the standard information of his/her avatar, his/her equipment and inventory, or his/her social relations in the game. This data could be used by the game server to propose some special customized game events to the players, such as a specific common quest requiring a particular object from two players' inventories.

This user model has been experimented in the prototype MugNSRC [8]. The original game, NSRC, is based on cartoon type wheelchair races in the office of a virtual Japanese Company. MugNSRC uses this context and integrates a user model with the player's motivation profile in the game engine as a mean to manage and develop a community through cooperative and competitive goals assigned to the players.

The question of which device hosts the user profile in the system relies on the global architecture of the game. Generally, multiplayer games follow a client/server architecture. The user profile is managed by the server, as in MugNSRC. Initial values of each class of data in the MUGPP are computed following the principle of a questionnaire. The player is invited to fill a form used to set the initial values of MUGPP parameters before the creation of his/her game account. These values could be changed according to a feedback loop related to the player choices and actions in the game. The user can log in to access his/her account, and retrieve his/her profile. On the other hand, P2P multiplayer games manage the player profile on the client side. The disadvantages of such an architecture are that the user has to manage himself/herself his/her profile when he/she changes to a new terminal, and that the players can cheat easily.

3. Smart Cards in the Management of User Profiles

Our work focuses on finding a way to manage efficiently player profiles in MUGs in order to provide a more personalized game to the gamers. Since network coverage and network connections are potentially unreliable, an interesting approach to carry out the game in a continuous manner would be to let the player carry his/her player profile along with him, so that the user is still able to play despite disconnection. This assessment leads to build a distributed and persistent information system for game data, and especially what we called MUGPP information. To manage this information, wearable devices are appropriate. The list of such devices includes mobile phones, PDA, Smart Cards, game consoles, memory cards, and so forth. Among those, Smart Cards are a good compromise in terms of wearability, security mechanisms, and costs.

Smart Cards are the most secure and widespread portable computing device today. They have been used successfully around the world in various applications involving money, proprietary data, and personal data (such as banking, pay-TV or GSM subscriber identification, loyalty, health-care, insurance, etc.). The Java Card [9] and the Microsoft.Net framework for Smart Cards are platforms that support a multiapplication environment, and in their modern versions, tend to go multithread. One of the key elements of Smart Cards is to improve on basic magnetic stripe cards with dynamically programmable microcontrollers, cryptographic Coprocessors, and means to protect the embedded data. Furthermore, Java Card platforms usually embed some code verifier, making those devices safer. Aside from their small size (to fit on a flexible plastic card and to increase hardware security) and from their low cost (to be sold in large volumes), this makes them ideal for any ubiquitous security-sensitive environment. Today Smart Cards are small computers, providing 8, 16, or 32 bits CPU with clock speeds ranging from 5 up to 40 MHz, ROM memory between 32 and 128 KB, EEPROM memory (writable, persistent) between 16 and 64 KB and RAM memory (writable, nonpersistent) between 3 and 5 KB. Smart Cards communicate with the rest of the world through Application Protocol Data Units (APDUs, ISO 7816-4 standard). The communication is done in client-server mode, the Smart Card playing the role of the server. It is always the terminal application that initiates the communication by sending a command APDU to the card and then the card replies by sending back a response APDU (possibly with an empty content).

Smart Cards can be accessed through a reader. The access has traditionally meant inserting the Smart Card in the reader. However, the trend is to interact in a contactless manner, to improve the Human Computer Interface (HCI) aspects. The Near Field Communication (NFC) technology provides devices with the ability to interact within a short range (less than 10 centimeters) by radio signal. This technology stems from the recent RFID market development. It works at a 13.56 MHz frequency, provides a 424 kbit/s bandwidth, and supports a half-duplex communication between devices. NFC Smart Cards combine the two previous technologies, so they are easily accessible in a contactless manner. Since these cards are non-self-powered, the radio signal from a reader is used to power the Smart Card-integrated circuit, in the same manner as RFID tags.

In the context of ubiquitous systems, the user can either carry an NFC Smart Card, which is readable within a short range by an NFC reader, or carry a reader, which is able to interact with the NFC devices disseminated over an area. As far as we know, there is no MUG that makes use of a Smart Card. However, there are some similarities between using a Smart Card for an MUG and using a Smart Card that is dedicated to commercial applications like public transportation systems and banking applications. Today, numerous cities in the world use contactless Smart Card-based systems to manage their public transportation system. For instance, the Paris commuters can use their contactless Smart Card (*Navigo*) as a mean to access transportation facilities (trains, buses, etc.) as well as the public bicycles network (*Velib*). The later involves a network of bicycle stations, which are equipped with an NFC readers, and a central authority, to help regulate the traffic. The Smart Card is used to store some user-related data, for example, the log of the stations he/she went through.

The core of this type of distributed information system is the management of user data on Smart Cards. There has been some effort to manage a health profile with PicoDBMS [10]. PicoDBMS is a Database management system dedicated to Smart Cards. PicoDBMS has also been used in some work undertaken by Lahlou and Urien [11] to filter some Internet data through a Smart Card-based user profile. They manage the profile dynamically (the user can specify his/her preferences). The security approach is that of the P3P (Platform for Privacy Preferences) [12] normalization group. The framework offers two security levels, the less secure being the less Smart Card intensive. The approach leaves out any gaming/ubiquitous aspect, and there is no mention of any authentication/confidentiality of the information. Ubiquitous systems should introduce a middleware to support this distributed information system. There are three essential components for these systems which are the users and their Smart Cards, the readers, and a central authority server.

4. Playing MUGs with NFC Smart Cards

The game system of some existing MUGs, such as [13–15], relies on the capability to control all the physical objects, which are integrated in the game, their impacts on the player, and all the various real-world embedded sensors, which take part in a hierarchy of networks. The participants of MUGs often experience the heavy load of physical wearable devices, or they have to deal with network disconnection problems [16]. Our proposal consists in using a Smart Card as an add-on interface for the interactions between the player, the virtual world, and the real world.

On the player's side, the MUGPP can be specified on a Smart Card, which enables the player to have access to some of his/her game-related information. The player can monitor his/her game process, manage his/her game objects, and even visualize or being informed with the game progress by either using one of the fixed terminals that are spread over the game area or by using a mobile terminal. In the context of an NFC Smart Card-based player profile, this would mean that the player could interact by using a Smart Card with a fixed NFC reader or with his/her mobile phone integrated reader.

The update of the MUGPP is executed automatically by the system and manually by the player. Firstly the MUGPP could be renewed by the player's physical interaction, that is to say, the player's physical movement and behavior in the real environment (outdoor and indoor). As the real environment is embedded with tangible objects, the player's physical location could be "tracked" as he/she walks through the game zones. The interaction between a Smart Card and a smart object using NFC readers can be performed without any connection to the game server. Every time the player comes close to a Smart Card reader, some of the MUGPP information can be updated and used in any way by dealing with the "as a player" data. Secondly, the MUGPP is updated following the communication or social interaction among players in the real world. The players should be able to sell and buy the game items they own to other players even while they are offline. The third group of information, that is to say, the "as his/her avatar" data can be updated dynamically. The social dimension of the gameplay is extended to the spatial and temporal dimension of the game. Therefore, the game system could trigger and control some game events in real time and real space for a group of players in the same game zone. Thus, the MUGPP can be updated during the real time interactions between the players, the game, and the physical space.

Playing MUGs with a Smart Card is a relatively new experience for the user, which will bring new forms of interaction to the players, new contents, and new security features.

Using a Smart Card gives the players new ways to interact with the game, potentially without any display device. This means that an automatic tangible interaction between the NFC Smart Card and the NFC reader can take place by bringing them close to one another. For the user, the most accessible and affordable mobile terminal is the mobile phone. Also, some are able to integrate the NFC technology, like the *Nokia 6131 NFC* and the *Sagem My700x*. Therefore, we suggest to use an NFC mobile phone to run a client application in our MUG system.

An ideal MUG is a digital environment with smart objects surrounding the user. This would allow him/her to interact with the game anywhere. Therefore, we can embed NFC readers in smart objects, such as Nabaztag [17], which could interact with each user's Smart Card. To enrich the user experience, a television decoder may also integrate an NFC reader so that the player could gain access to the multimedia content related to the game.

On the Smart Card, we aim at defining and formalizing an MUGPP which might help maintain decentralized user data from the game server. This MUGPP allows the user's personal information to be reused by several game mechanisms and to be completed by several applications. The interest of having an MUGPP on a Smart Card is not only that users have a more "wearable" computing device but also that the game designers can provide each individual with a personalized gaming experience. In the mechanism of a MUG, the MUGPP can take a central role rather than being a peripheral or real context to influence the game server in making the decision for a customized service to the end user.

Considering security aspects, the specification of player profiles as separated from the server will guarantee the confidentiality of each individual's private information and the related service. For example, it could be possible to register the information of the player's bank account on the Smart Card which allows the player to have access to a paying service. In "World of Warcraft" (Blizzard Entertainment, 2004), the user can register his/her bank account on the game server, which can be unsafe despite the login/password protection, in order to obtain some special services from the game editor. From a perspective point of view, this will enlarge the possibility of license management such as biological or vocal based identity.

As a consequence, there is a need to support Smart Card, NFC reader in the MUG system architecture. We will describe an API which provides this service in more details in the following sections.

5. Architecture to Manage MUGPP on an NFC Smart Card

The NFC interactions in MUGs (see Section 4) and of the MUG player profile (see Section 2) are key issues of our proposal. The main component of this architecture is the service that manages the MUG player profile on the external NFC Smart Card. We have implemented a library which enables *Java 2 Micro Edition* [18] (J2ME) *Mobile Information Device Profile-* (MIDP-) based mobile phones to exchange data with Smart Cards and game server logic. The server is itself implemented in J2SE and the Smart Card part of the application is a Java Card cardlet. Finally, we use the security mechanisms to ensure the privacy of the player profile data. Figure 1 presents an overview of the MUGPPM architecture.

5.1. Oncard Service. Our card-side implementation aims the card applications based on Java Card platform which complies with the ISO 14443 [19] standard part 1, 2, and 3 type A. An oncard Java applet is dedicated to the MUGPPM. It implements a set of instructions to handle communication with an NFC reader. These instructions are built using the Application Protocol Data Unit (APDU) protocol defined in the ISO/IEC 7816 standard. Besides, it maintains the player profile data model with some added security features.

5.1.1. APDU Instructions. The APDU instruction set used in MUGPPM allows the following:

(i) to manage the player profile,

(ii) to manage the default entries, for example, static entries of our MUG player profile definition, for example, the *username*, *age,* or *playtime* fields,

(iii) to manage object entries, for example, entries corresponding to game data objects, like inventory items,

(iv) to manage the private and public key entries.

With this set of instructions, the reader can access a profile stored on the Smart Card, save/load each profile fields independently, and store/retrieve the game objects. Objects can be defined as exchangeable between players. Nevertheless, it is the MUG game designer who has to decide if a game object is sharable or not. Table 1 shows the instructions used by the MUGPPM. It details the parameters of each instruction and the corresponding response of the Smart Card.

5.1.2. Data Model. The field lengths have been bounded due to the memory limitation that characterizes the Smart Card platforms. We tested our implementation on an *Oberthur Cosmo Dual* card which offers only 72 KB of memory space (EEPROM).

Nevertheless the profile itself is not really heavy, since the CAP file containing our oncard application uses around 6 KB. We also use a 4 KB memory buffer to deal with large I/Os. The fields of the GameProfile class themselves include a number of byte arrays (264 bytes), and a couple of OwnerPIN objects to manage the user password and the game provider password (the object size depends on the Java Card Virtual Machine (JCVM) implementation, but the password itself is limited to 8 bytes). Furthermore, the profile is associated with three 2048 bits RSA keys (768 bytes). So the application itself requires 8 KB and each instance of a profile should require less than 2 KB (depending on the JCVM implementation). Therefore, we could theoreticly manage about 30 different game profiles with this Smart Card.

5.2. NFC Reader Side Service. The main functionalities of the NFC reader API are to access the MUG player profile stored on the Smart Card and to communicate with the profile-based services that are hosted on the MUG server.

The APDUDataManager class is used to establish the NFC communication toward the card and to some send APDU formatted messages. The GameProfile class is used to manage the player profile fields during profile manipulation on the reader. Finally, the NetworkCom class handles object-oriented HTTP communications with the server based on the MooDS protocol [20].

We have prototyped a J2ME version of our MUGPPM service to have a Java mobile phone access to the oncard MUGPPM service. This choice is obvious considering the mobile phone is the most widespread mobile terminal for end-users. Moreover, in 2007, some J2ME mobile phones embedding NFC readers, such as the *Nokia 6131 NFC* or the *Sagem my700X*, were placed on the market. An API to help establish a contactless communication between a J2ME mobile phone and an NFC Smart Card has been released the same year: the *JSR257* [19].

A specific API is traditionally used to handle an APDU-based communication on J2ME mobile phones: the *JSR177* [21]. However, the use of this API is not mandatory in the case of an external NFC Smart Card. In fact, it offers essential mechanisms enabling the mobile phone to communicate with its embedded SIM card. Thus, our prototype uses the JSR257 functionalities to initiate a communication between the mobile phone and the Smart Card.

In order to use the MUGPPM functionalities, the first step for the player is creating his/her MUGPP on the Smart Card. So, he/she has to enter a login and a password which will be used to access his/her profile. In this first step, he/she has to enter his/her personal information, for example, the user "by himself" is part of the MUGPP. Thus, the API can load the player profile fields from the card onto the mobile phone, to store them in the profile object representation. Afterward, the MUG client game engine can start using the player profile as it is defined in the MUG game design. Figure 2 summarizes the architecture used to provide the MUG client game engine with an access to the oncard MUG Player Profile Management service.

It is important to notice that the interaction between the mobile phone and the NFC Smart Card depends on the player since, he/she has to draw the card near the mobile phone during all the process, for example, during a game save. The MUG game designer must take into account this specific Human Computer Interaction (HCI).

Besides, our prototype can communicate with an MUG HTTP server. It uses the MooDS protocol to communicate

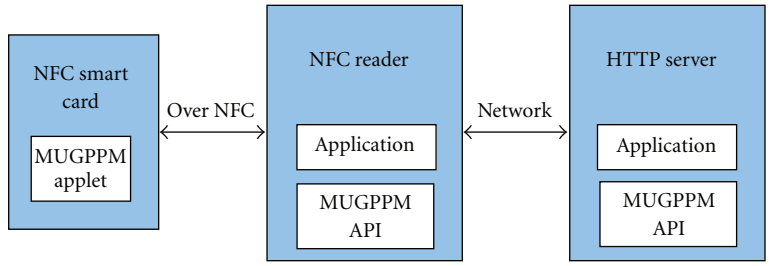

Figure 1: MUGPPM architecture overview.

Table 1: APDU instructions used in MUGPPM.

Instruction	P1	P2	Data	Returns
CREATE_PROFILE			login+pwd	status
LOGIN_PROFILE			login+pwd	status
REINIT_PROFILE			login+pwd	status
DELETE_PROFILE			login+pwd	status
LOAD_DEFAULT_ENTRY	key			data
UPDATE_DEFAULT_ENTRY	key		data	status
LOAD_OBJECT_ENTRIES				data
LOAD_OBJECT_ENTRY	key			data
ADD_OBJECT_ENTRY	key	isSharable	data	status
DELETE_OBJECT_ENTRY				status

with the MUG server in an object-oriented manner. The developer can creates objects which represent the messages used during the client-server communication. Thus, if the MUG client needs a profile-based service from the server, it has to instantiate the corresponding message object and send it through the MooDS encoder. We have created a message to invoke a server side service, the ProfileBasedServiceRequest message class. It can also decode the server response using the MooDS decoder and handle the decoded message objects. If the service requires data from the Smart Card, the client receives a CardDataRequest message from the server which contains a list of required field keys. Then, the MUGPPM API can retrieve the associated fields data from the Smart Card and send it within a DataCardResponse object to the server. Finally, it receives the service response, for example, a player list from a lobby service.

5.3. Server Side Service. The MUGPPM server API offers a Java based MUG server the ability to create a profile based service. It helps create personalized services, for example, profile based lobby or profile based quest provider. The server API and the client API have a similar class to handle MUGPP contents: theGameProfile class. For example, if the server requires the player nationality, it has to request the corresponding field key from the Smart Card and to handle the card response. The communication part of the API is also based on the MooDS protocol.

To request a service, the client has to send a ProfileBasedServiceRequest message with the name of the service needed. Then, if the service requires personal data stored on the Smart Card, it sends back a CardDataRequest message to the client containing a list of required field keys. Afterwards, it receives from the client a DataCardResponse message which contains the required data. Finally, the service computes the response based on the received player personal data and returns a specific response message to the client.

5.4. MUGPP and Security. Some of the MUGPP data deal with the user private life. Furthermore, the lack of a sound and secure authentication procedure typically makes cheating in MUGs an easy feat [22, 23]. There is a need to use improved security mechanisms to act against those threats.

The players and the terminal the players use (in our case a mobile phone) are by definition untrusted, but the oncard application can be securely and reliably developed using Java Card.

In order to insure the security of the player's private data, the card requires an authentication from the reader. This authentication process is based on a personal login/password chosen by the player during his/her account creation. We use the OwnerPIN class on the card to safely store the user password. The login procedure needs to be performed to authorize the access to the smart card cryptographic functionalities. When the user is not playing any more, the user is logged out from the Smart Card. The application provider uses another PIN code to block/unblock the user from modifying certain fields.

We chose to use a public key infrastructure to help the MUG system designers ensure the security of the application. Yet, the management of the keys on a Smart Card is a non trivial issue. The Smart Card requires a personalization phase during which a key pair is created and stored on the

Figure 2: MUGPPM architecture for J2ME devices.

card static memory. The server side also requires a key pair, and an X.509 infrastructure is used to certificate the use of public keys. This public key infrastructure guarantees the privacy of communications between the server and the Smart Card. Thus, when user logs in, he/she does so, not with a Smart Card, and not with a server. He/she can then have access to a higher security level than just a password-based protocol.

When the application needs to interact with the server, the server sends its public key as well as a certificate. The Smart Card can then verify the validity of the key. If the key happens to be valid, the Smart Card can keep the public key. The Smart Card can send its public key to the server. All subsequent interactions between the server and the client can then use an encryption/decryption using one's private key and the other's public key.

The overall mechanism guarantees a stronger identification scheme than just a login/password and might help thwart some common online games cheats. One advantage here is that no critical data is transmitted in plain text format over the network.

A common cheat is the replacement of code or data concerning the game. The simple fact of using a Smart Card to manage the MUGPP makes it considerably difficult to tamper with the game profile, the cheater being unable to directly hack into the profile/oncard game infrastructure. The game designer might want to check an additional server signature for any operation that modifies some elements in the profile.

An other cheat consists in abusing the game procedures. For instance, a player can log out before he/she loses a game. Making the signing of some game procedures by the server necessary can be used as a countermeasure against such cheats.

The mobile aspect of our framework implies that some interactions between two players can occur out of a connection with the game server. For instance, in a role-playing game, the players might want to exchange an item. This operation could take place without a server while still guarantying the nonrepudiation property.

6. MUG PPM Use Cases

Our library can be used for different types of interactions, *connected* or *disconnected* interactions from an MUG server point of view. For example, the secure architecture of the MUGPPM can only be used safely with a network connection in order to validate public keys with a signing authority through a registered MUG server. So, secure interactions have to be carried out in a connected way. However, disconnected interactions are possible without strong security mechanisms, particularly for local interactions. Thus, an MUG can introduce NFC checkpoints or local object exchange mechanisms between players using this API.

6.1. Connected MUG Interaction Examples. Our framework can be used to provide various profile-based connected services in a secure way, like providing players with personalized quests or locating players who speak a common language in a game area.

Via mineralia [24] is a pervasive search and quizz game in the museum of Terra Mineralia in Freiberg. The goal of the game is to realize quests in the context of the mineral exposition. Each point of interest is represented by an RFID tag on the mineral. The MUGPPM can be used in this application to check the visitor card at the museum entry (with an NFC reader) to adapt game content to his/her player profile. For example, different levels of mineral knowledge could be set to fit the category of the visitor (novice, expert, etc.) and to propose personalized quests. Moreover, regular visitors could resume a quest undertaken previously.

As an another use case, we have implemented a profile-based lobby service on top of the MUGPPM secure architecture. This service uses the player's age and the languages he/she knows. The server asks for the user's required personal data while using the security part of MUGPPM. Finally, the profile based lobby service computes the list of connected players matching the required age and spoken languages and returns it to the client. That type of service could have been used in games like the item hunt game "Mogi Mogi" [15]. In this game, some users have been using a lobby-like

application to spy on other younger players. Bypassing the game rules this way can be controlled using our API. Indeed, as the private data is stored on a secure decentralized device (unlike a game server), fraudulent use of personal data is rendered more difficult, while statistics can still help detect that type of behavior.

6.2. Disconnected MUG Interaction Examples. MUG game designers can integrate disconnected interactions in their game by using the MUGPPM API.

Paranoia Syndrome [25] is a classic strategic game that integrates some location based interactions, and RFID tangible objects. One of the perspectives of the game, is that multimedia content and basic AI will be added to the tangible objects to serve different content by regarding the player type (doctor, scientist, alien, etc.). With MUGPPM, the interactive objects (with an embedded NFC reader) could adapt their content and interaction to the player with regard to the player profile in a disconnected way.

Furthermore, a MUG can integrate difficulty levels corresponding to the player's age in order to assign a course to the player in the game area. This interaction can be made between the player and a NFC checkpoint and does not necessarily require a server side resolution.

In addition, MUGs can implement game object exchange mechanisms between players. Such a service should give two players in the same *real world* area, the ability to exchange some game items from their inventories. This interaction can be made by peering the mobile phones of the players over a local communication link. The NFCIPConnection class from the com.nokia.nfc.p2p package (available in the Nokia JSR257 implementation) allows to establish a NFC link between two phones. We have implemented a game object exchange service, on top of our API, that offers to a player to send one sharable item from his/her game inventory to another player. We consider here that each player has previously loaded his/her player profile from the Smart Card. This list can be retrieved from the object representation in the player profile (see Section 5.2 for more details about the profile loading mechanism). So, a player who wants to send an object to his/her friend has to select the item from his/her list and the sender mode, whereas the other player has to select the receiver mode. The players must approach their mobile phones in order to set up the P2P link. As soon as the connection is established, the object is sent as a byte array onto the network. Then, the receiver handles the binary data corresponding to the item and can add it to his/her inventory. Finally, the new inventories of both players will be updated in the Smart Card during their next game save.

These examples emphasize a major benefit provided by our API in the MUG domain: it does not require the players to be connected with the central MUG server in order to interact in the game. Thus, our library enables new interactions for MUG in a totally decentralized manner.

To evaluate the performance of our application, we used the Mesure project [26], which is dedicated to measuring the performance of smart cards. The Mesure project provides detailed time performance of individual bytecodes and API calls. Given the use cases described earlier, we monitored the use of each bytecode and each API call for a regular use of our application. We then matched the list of used bytecodes and API calls with the individual performance of each feature measured on our smart card. The results show that the time necessary to perform a RSA encryption with the smart card is close to half a second, and it is by far the costliest of the operations described earlier. Login into the smart card, as a title of comparison lasts less than 20 milliseconds.

7. Conclusions and Perspectives

This paper presents an NFC Smart Card based approach to handle the player profile in the context of MUGs. This NFC card centric architecture allows new kinds of interactions in both centralized and decentralized ways. The main advantage of our method is to allow the players to play at any time, and anywhere, hence the ubiquitous aspect of the game. We have presented the MUGPPM API which is dedicated to the Java Card/J2ME/J2SE platforms. This enables MUG developers to implement a Smart Card based architecture to provide profile-based services. Thus, players can have a personalized game experience. Besides, this API provides the player with a secure way to ensure a certain level of data confidentiality. We will release the MUGPPM server API as an open source OSGi bundle to be integrated in the uGASP [2, 3] middleware. Thereafter, game developers could implement MUGs based on this framework, therefore offering personalized services.

On the basis of our framework, it is possible to specialize and realize an authoring tool for the development of MUGs. It would be interesting to consider using the NFC Smart Card from a more conceptual point of view during the design of the game. Using Smart Cards in MUGs may also give rise to the future direction of game design by developing new forms of interaction and narration based on new technology of mobility and ubiquity.

The question of "who personalizes the Smart Card" remains open. In traditional banking, telecom or transport applications, this is carried out by the card emitting company. However, the ongrowing multiapplication aspect of Smart Card makes it more and more questionable. For the purpose of testing our API, we let the user fill out the form which might be questionable for a secure application. Still, the application provider has some control over the fields through its own PIN code.

Future works include a generalization of the security architecture in terms of key sizes and algorithms, depending on the functionalities of a given Smart Card.

In addition, we will generalize the API to facilitate the description of services and to manipulate the player profile data structure. On the server side, this should help the describe connected the player profile-based services. On the card side, we will investigate PicoDBMS database to handle the player profile data structure.

We await the results of an other project: T2TIT [27] (Things to Things in the Internet of things). This project proposes to interact with contactless object, going as far as to give them a network identity, while keeping some strong security properties. The eventual conclusion of T2TIT can be helpful to us, for instance, we can expect to use some

encrypted channels. We intend to use the T2TIT security mechanisms in our work. The newly published Java Card 3.0 specification [9] introduces multithreading mechanisms in Smart Cards. This suggests other interactions between different profiles, which were not considered in this paper.

In terms of oncard code verifiers, works like embedded data flow analysis (see [28]) might also provide us with some strong on card inter-application protection features. We could reliably share some data from one profile to an other, and deny the access to such data from other profiles.

We have not explored here the issues of biometric identification. It is clearly complementary to the traditional cryptographic schemes, and as the Smart Card industry is integrating more and more of those, so should we.

References

[1] S. Björk, M. Börjesson, P. Ljungstrand, et al., "Designing ubiquitous computing games—a report from a workshop exploring ubiquitous computing entertainment," *Personal and Ubiquitous Computing*, vol. 6, no. 5-6, pp. 443–458, 2002.

[2] R. Pellerin, E. Gressier-Soudan, and M. Simatic, "uGASP: an OSGi based middleware enabling multiplayer ubiquitous gaming," in *Proceedings of the International Conference on Pervasive Services (ICPS '08)*, Sorento, Italy, July 2008, Demonstration Workshop.

[3] GASP/uGASP project, http://gasp.ow2.org.

[4] OSGi alliance, http://www.osgi.org/Main/HomePage.

[5] S. Natkin and C. Yan, "User model in multiplayer mixed reality entertainment applications," in *Proceedings of the ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE '06)*, Hollywood, Calif, USA, June 2006.

[6] NFC Forum, http://www.nfc-forum.org/home.

[7] The PLUG project, http://www.capdigital.com/plug/.

[8] C. Yan, *Adaptive multiplayer ubiquitous games: design principles and an implementation framework*, Ph.D. thesis, Cotutelle Research Program with Orange Labs and CNAM, Paris, France, 2007, Supervisor: Stephane Natkin.

[9] Java Card platform, http://java.sun.com/javacard.

[10] P. Pucheral, L. Bouganim, P. Valduriez, and C. Bobineau, "PicoDBMS: scaling down database techniques for the smartcard," *Very Large Data Bases Journal*, vol. 10, no. 2-3, pp. 120–132, 2001.

[11] A. Lahlou and P. Urien, "SIM-Filter: user profile based smart information filtering and personalization in smartcard," in *Proceedings of the Ubiquitous Mobile Information and Collaboration Systems (UMICS '03)*, Klagenfurt/Velden, Austria, June 2003.

[12] Platform for Privacy Preferences (P3P) Project, http://www.w3.org/P3P.

[13] S. Jonsson, A. Waern, M. Montola, and J. Stenros, "Game mastering a pervasive larp. Experiences from momentum," in *Proceedings of the 4th International Symposium on Pervasive Gaming Applications (PerGames '07)*, Magerkurth, Carsten, et al., Eds., pp. 31–39, Salzburg, Austria, June 2007.

[14] O. Sotamaa, "All the world's a botfighter stage: notes on location-based multi-user gaming," in *Proceedings of the Computer Games and Digital Cultures Conference (CDGC '02)*, F. Mäyrä, Ed., Tampere, Finland, June 2002.

[15] Mogi Mogi,http://www.mogimogi.com.

[16] D. Cheok, et al., "Human Pacman: a mobile, wide-area entertainment system based on physical, social, and ubiquitous computing," *Personal and Ubiquitous Computing*, vol. 8, no. 2, pp. 71–81, 2004.

[17] Friedrich von Borries, Steffen P. Walz, and Matthias Böttger, "Mogi: Location-Based Services—A Community Game in Japan ," in *Space Time Play*, vol. 2007, pp. 224–225, Birkhäuser Basel, Switzerland, 2008, http://www.springerlink.com/content/j0277056ult42551.

[18] J2ME MIDP, http://java.sun.com/javame/index.jsp.

[19] JSR257, http://jcp.org/en/jsr/detail?id=257.

[20] R. Pellerin, "The MooDS protocol: a J2ME object-oriented communication protocol," in *Proceedings of the 4th Mobility Conference*, Singapore, September 2007.

[21] JSR177, http://jcp.org/en/jsr/detail?id=177.

[22] J. Yan and B. Randell, "A systematic classification of cheating in online games," in *Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames '05)*, New York, NY, USA, October 2005.

[23] N. E. Baughman, M. Liberatore, and B. N. Levine, "Cheat-proof playout for centralized and peer-to-peer gaming," *IEEE/ACM Transactions on Networking*, vol. 15, no. 1, pp. 1–13, 2007.

[24] G. Heumer, F. Gommlich, B. Jung, and A. Müller, "Via Mineralia: a pervasive museum exploration game," in *Proceedings of the 4th International Symposium on Pervasive Gaming Applications (PerGames '07)*, pp. 157–158, June 2007.

[25] G. Heumer, D. Carlson, S. H. Kaligiri, et al., "Paranoia Syndrome: a pervasive multiplayer game using PDAs, RFID, and tangible objects," in *Proceedings of the 3rd International Symposium on Pervasive Gaming Applications (PerGames '06)*, pp. 157–158, June 2007.

[26] The Mesure project, http://mesure.gforge.inria.fr.

[27] P. Urien, et al., "The T2TIT research project. Introducing HIP RFIDs for the IoT," in *Proceedings of the 1st International Workshop on System Support for the Internet of Things (WoSSIoT '07)*, Lisbon, Portugal, March 2007.

[28] D. Ghindici, G. Grimaud, and I. Simplot-Ryl, "An information flow verifier for small embedded systems," in *Proceedings of the International Workshop on Information Security Theory and Practices (WISTP '07)*, vol. 4462 of *Lecture Notes in Computer Science*, pp. 189–201, May 2007.